Appl. No. 10/621,951
Response to 1st Action dated 06/01/2006
Reply to Office Action of 03/01/2006

REMARKS

In the above-identified Office Action, the Examiner objected to Figs. 11, 12, 13 and 15 of the Drawings, to the Specification and to Claims 10 and 17. Claims 1, 8, 11, 13 – 15, 18 and 20 were rejected under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski. Claims 11 and 13 were rejected because they are drawn to computer programs claiming the same invention as system Claims 18 and 20. Claims 2, 4, 6, 7, 9 and 16 were rejected under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski and further in view of Achiwa et al. Claims 9 and 16 were rejected because they are drawn to a program and system claiming the same invention as method Claim 2. Claims 10 and 17 were rejected under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski in view of Achiwa et al. and further in view of Bauer. Claim 10 was rejected because it is drawn to a computer program claiming the same invention as system Claim 17. Claim 3 was rejected under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski in view of Achiwa et al. in view of Nevarez and further in view of Bauer. Claims 12 and 19 were rejected under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski and further in view of Falkner. Claim 5 was rejected under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski in view of Achiwa et al. and further in view of Falkner.

In response to the objection to the DRAWINGS and to the SPECIFICATION, Applicants have amended the Specification to include the reference numerals which the Examiner said are in the DRAWINGS but not in the SPECIFICATION. Applicants have further amended the SPECIFICATION to delete the word "as" as suggested by the Examiner. Due to these amendments to the Specification, Applicants submit that the objections to the DRAWINGS and SPECIFICATION have been overcome. Hence, Applicants respectfully request withdrawal of those objections.

Applicants have also amended independent Claims 1, 8, 15 to include the limitations of "determining at least one frequently-accessed file system object in a file system upon mounting the file system at a mount point on a computer system." Based on this amendment, the 112 rejection made to Claims 10 and 17 is been overcome. Consequently, Applicants kindly request withdrawal of this rejection.

By amending the independent claims as shown above, original Claims 2, 9 and 16 became superfluous and are thus amended to recite the limitations of "the pathname being relative to the mount point" and to being dependent on Claims 3, 10 and 17, respectively. Support for the added limitations can be found on page 11, lines 15 – 18 and Fig. 8.

Claims 3 - 7, 10 and 17 are amended to change their dependency from Claims 2, 9 and 16 to Claims 1, 8 and 15, respectively. The claims are also amended to better claim the invention.

The Examiner rejected Claims 9, 10, 11, 12, 13 and 16 because they are drawn to computer program product claiming the same invention as some system claims. Applicants do not understand the rationale of this rejection and kindly request that the Examiner provide sources, authorities etc. in support of this rejection.

By this amendment, Claims 1 - 20 remain pending in the Application. For the reasons stated more fully below, Applicants submit that the pending claims are allowable over the applied references. Hence, reconsideration, allowance and passage to issue are respectfully requested.

As stated in the SPECIFICATION, in Unix file systems, a directory is considered to be a file and each file is associated with an index node or inode. An inode is a data structure that contains important information about the file with which it is associated. Information contained in an inode includes user and group ownership of the file, access permissions (e.g., read, write, execute permissions) and file type (e.g., regular, directory or device file). Further, the inode contains the date and time the file was created as well as the date and time of any modifications. In addition, the inode contains information regarding the location of the file on a disk or storage system. The inode is identified by a unique number called an inode number. Thus, to access a file on a disk, the file inode number must be known.

Users, however, do not access files using the files' inode numbers; rather, they use the files' symbolic names. Hence, a table is used in which files' symbolic names are cross-referenced with their inode numbers. This table is generally referred to as a directory.

Symbolic names are often in terms of pathnames. To obtain the inode number of a file referred to by its pathname (e.g. /usr/lib/libc.a), a plurality of steps may occur. Particularly, the inode number of each element or "edge" of the pathname (e.g., usr, lib, libc.a) has to first be obtained from its parent directory.

Thus, the system must perform a lookup of the first edge of the pathname within its parent directory (either the root directory for an absolute pathname or the current directory for a relative pathname). First, the contents of the parent directory are examined to cross-reference the edge name with its inode number. Next, using the inode number found, the inode is accessed and if the user has sufficient access permission and the edge is a directory, then a name lookup is performed of the next edge of the pathname in the directory just found. This process is repeated until the inode number for the last edge is determined, which is then returned to the process or thread performing the name lookup.

Since directories are stored on disks, each inode access is a disk access. Generally to open a file, multiple disk accesses are required for every edge in the pathname. Particularly, one disk access is used to look up the edge name in the parent directory, another is used to access the inode for the edge and at least one more is used to access the content of the object being looked up. It is well understood in the art that disk accesses are more time-intensive than memory accesses. Thus, to increase performance, a directory name lookup cache (DNLC) is used.

The DNLC is a general file system service that caches the most recently referenced file names and their associated inode numbers. Thus, the DNLC can satisfy any subsequent request for any of the information contained therein. Therefore, when an application (e.g., a text editor or a compiler) tries to look up a file name or requests file data, the DNLC is first checked for the name of each directory/subdirectory or file in the pathname of the file. If the name is in the DNLC, the inode number will be obtained.

However, just as in the case of the disk, each name lookup is a DNLC access. Thus, in cases where a particular file is used by a plurality of other processes such that it is consistently being opened and closed, the system will access the DNLC as often as it would the disk in order to find the location of the file on the disk.

Hence, since each DNLC access consumes time, although not to the same extent as a disk access, it would therefore be advantageous to decrease the overhead associated with frequent lookups of particular pathnames.

The present invention provides a method that decreases the overhead associated with frequent lookups of particular pathnames. According to the teachings of the invention, when a file system containing file system objects (e.g., files) that are to be frequently accessed by a computer system is mounted onto the computer system, the pathnames of the file system objects are cross-referenced with their inode numbers and entered into a memory system (e.g., a cache) on the system. This decreases the overhead associated with accesses to the files since one access to the memory access will provide the inode number, which is what is needed to locate the file system object on a storage system.

The step of cross-referencing the pathname of an object to its inode and storing it in the memory access upon mounting the file system on a computer system further increases the performance of the system as the inode of the object will be obtained from the memory system upon the first time the object is being accessed rather than after the object has been accessed a certain number of times.

The invention is set forth in claims of varying scopes of which Claim 1 is illustrative.

> 1. A method of providing a performance-enhancing way of accessing frequently-accessed file system objects comprising the steps of:
> ***determining at least one frequently-accessed file system object in a file system upon mounting the file system at a mount point on a computer system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system***;
> entering the pathname of the at least one file system object into a memory system; and
> cross-referencing the pathname of the at least one file system object in the memory system with its inode number thereby enabling the inode number to be obtained with one memory access. (Emphasis added.)

The Examiner rejected the claims under 103 as being unpatentable over Sinha and Pinkowski. Applicants disagree.

Sinha purports to teach a pathname resolution method for providing fixed speed of file accessing in computer network. According to the purported teachings of Sinha, each user of a node of a distributed system can selectively specify one of a plurality of modes of path name resolution for use in accessing a specific file from a node of the system. The selected mode provides a fixed, minimum access time for the file and utilizes a name cache within the main memory of that node.

In one of the selected modes, the name cache contains pathnames of frequently accessed files which are cross-referenced to the location of the files. For example, if the file is on a remote system, the location of the file will contain an identification of the remote system and information to locate the file on that remote system.

However, Sinha does not teach, show or suggest the step of ***determining at least one frequently-accessed file system object in a file system <u>upon</u>***

mounting the file system at a mount point on a computer system. Rather Sinha teaches only that frequently accessed files, which are determined with the use of a counter for counting how many times the file is accessed, are entered in the name cache.

Pinkowski, on the other hand, purports to teach a method and apparatus for file server addressing. According to the purported teachings of Pinkowski, a file handle in a call that accesses a file includes a file system identification, a file identification or inode number and a generation number. The addressing system converts the multi-bit file identification number into an alternative path name that identifies a location for the file without the need for converting the real path name. Specifically, the inode number in binary form is translated into a multiple digit hexadecimal form that is parsed into directory names. The last directory name provides the location of the designated file.

However, just as in the case of Sinha, Pinkowski does not teach, show or suggest the step of ***determining at least one frequently-accessed file system object in a file system upon mounting the file system at a mount point on a computer system***.

Achiwa et al. purport to teach a method for file space management. According to Achiwa et al., the method provides for moving one or more files from a client storage system to a server storage system. Specifically, a file from the client storage system is copied to the server storage system and is deleted from the client storage system, thus recovering storage space in the client storage system. A logical reference is provided in the client storage system to allow access requests to be made to the file from the client storage system, even though the file has been deleted therefrom. The logical reference also allows for the file to be reproduced in the client storage system when needed.

However, as with Sinha and Pinkowski, Achiwa et al. do not teach, show or suggest the step of ***determining at least one frequently-accessed file system object in a file system upon mounting the file system at a mount point on a computer system***.

Note that the other applied references (i.e., Nevarez, Bauer and Falkner) do not show the above-emboldened/italicized limitations either. Hence, Applicants submit that Claim 1, as well as its dependent claims, should be allowable. Independent Claims 8 and 15, which all incorporate the above-emboldened-italicized limitations in the above-reproduced claim 1, together with their dependent claims, should also be allowable.

Consequently, Applicants once more respectfully request reconsideration, allowance and passage to issue of the claims in the application.

Respectfully Submitted

By: [illegible]
Volel Emile
Attorney for Applicants
Registration No. 39,969
(512) 306-7969